第九章 关系查询处理和查询优化

沈明玉

合肥工业大学 Hefei University of Technology 计算机与信息学院

第9章 关系查询处理与查询优化

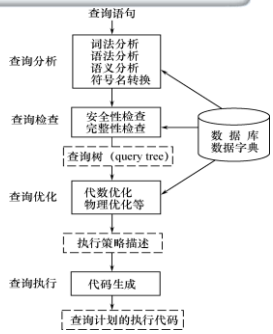
- 查询处理：
  DBMS执行用户查询请求的过程。
- 查询优化：
  DBMS提高查询执行效率的手段。
- 用户通过客户端数据接口，借助网络平台向数据库服务器发送查询请求（SQL语句），数据库服务器会产生相应的服务进程来处理用户的查询请求。
- 本章内容就是介绍服务器的服务进程是如何处理并优化用户的查询请求的。
- 在查询处理和查询优化的过程中涉及到一些技术实现，如数据库的安全性控制、完整性控制等，甚至还与数据库物理设计的内容有关。

第9章 关系查询处理与查询优化

9.1 关系数据库系统的查询处理

9.1.1 查询处理步骤

- **查询处理的四个阶段：**
  - 查询分析
  - 查询检查
  - 查询优化
  - 查询执行



9.1 关系数据库系统的查询处理

- **查询分析：**借助数据字典分析语句是否可以被执行
  - ✓ 对查询语句进行扫描、词法分析和语法分析 ；
  - ✓ 从查询语句中识别出语言符号，如语言符号、属性名、关系名等，进行语法检查和语法分析，即判断查询语句是否符合SQL语法规则 ，若正确，则进行下一步的查询检查，否则终止处理并返回错误代码。
- **查询检查：**从安全性、完整性角度检查是否允许用户的查询请求
  - ✓ 根据数据字典对合法的查询语句进行语义检查；
  - ✓ 检查用户存取权限、完整性约束等，若用户权限不足或违反完整性约束规则，则终止处理，并返回错误代码，否则继续后续处理；
  - ✓ 将SQL语句转换成等价的关系代数表达式（语法分析树）；
  - ✓ 把数据库对象的外部名称转换为内部表示 。

9.1 关系数据库系统的查询处理

- **查询优化：**如何更有效地执行用户的语句
  - ✓ 查询优化：选择一个高效的查询处理策略
  - ✓ 查询优化分类：
    - 代数优化：指关系代数表达式的优化
    - 物理优化：指存取路径和底层操作算法的选择
  - ✓ 查询优化方法选择的依据：
    - 基于规则(rule based)
    - 基于代价(cost based)
    - 基于语义(semantic based)

9.1 关系数据库系统的查询处理

- **查询执行：**最优策略的执行
  - ✓ 依据优化器得到的执行策略生成查询计划；
  - ✓ 代码生成器(code generator)生成执行询计划的代码；
  - ✓ 执行代码，返回查询结果及状态。

9.1.2 实现查询操作的算法示例

1. 选择操作的实现

［例1］Select * from student where <条件表达式>；

考虑<条件表达式>的几种情况：

C1：无条件；

C2：Sno＝'20160001'；

C3：Sage>20；

C4：Sdept＝'CS' AND Sage>20；

(1) 简单的全表扫描方法

① 按照物理次序读Student的M块（假定可用的内存为M块）到内存；

② 检查内存中的每个元组t，如果t满足选择条件，则输出t；

③ 如果Student还有其他块未被处理，重复①和②。

✓ 适合小表，一般不适合大表

✓ 内存要求小（最小1块）

（2）索引扫描法

✓ 适合选择条件中的属性上有索引(例如B+树索引或Hash索引)

✓ 通过索引先找到满足条件的元组主码或元组指针，再通过元组指针直接在查询的基本表中找到元组

[例] 以C2为例，Sno＝'20160001'，并且Sno上有索引，使用索引得到Sno为'20160001'元组的指针，通过元组指针在student表中检索到该学生。

[例] 以C3为例，Sage>20，并且Sage上有B+树索引，使用B+树索引找到Sage＝20的索引项，以此为入口点在B+树的顺序集上得到Sage>20的所有元组指针，通过这些元组指针到student表中检索到所有年龄大于20的学生。

[例] 以C4为例，Sdept＝'CS' AND Sage>20，如果Sdept和Sage上都有索引：

算法一：分别用上面两种方法分别找到 Sdept＝'CS' 的一组元组指针和 Sage>20 的另一组元组指针，求这两组指针的交集，到student表中检索，得到计算机系年龄大于20的学生。

算法二：找到Sdept＝'CS'的一组元组指针，通过这些元组指针到student表中检索，对得到的元组检查另一些选择条件(如Sage>20)是否满足，把满足条件的元组作为结果输出。

结论：考虑到索引操作的开销，只有当大表且选择率较低，适合用索引扫描法，否则适合用全表扫描。

2. 连接操作的实现

连接操作是查询处理中最耗时的操作之一，本节通过简单示例了解等值连接(或自然连接)最常用的几种实现算法。

SELECT * FROM Student，SC　WHERE Student.Sno=SC.Sno；

（1）嵌套循环方法

算法思想：对外层循环(Student)的每一个元组，检索内层循环(SC)中的每一个元组，若这两个元组在连接属性(sno)上相等，则串接后作为结果输出，直到外层循环表中的元组处理完为止。

外层表Student扫描一次，内层表SC的扫描次数取决于Student的元组数。

（2）排序-合并方法

适合连接的诸表已经排好序的情况。

排序-合并连接方法的步骤：

✓ 如果连接的表没有排好序，先对Student表和SC表按连接属性Sno排序；

✓ 取Student表中第一个Sno，依次扫描SC表中具有相同Sno的元组；

✓ 当扫描到Sno不相同的第一个SC元组时，返回Student表扫描它的下一个元组，再扫描SC表中具有相同Sno的元组，把它们连接起来；

✓ 重复上述步骤直到Student 表扫描完。

要考虑表的排序开销，当两个大表连接时，该方法是有效的。

9.1 关系数据库系统的查询处理

（3）索引连接方法

算法步骤：

① 在SC表上建立属性Sno的索引。

② 对Student中每一个元组，由Sno值通过SC的索引查找相应的SC元组。

③ 把这些SC元组和Student元组连接起来

循环执行②③，直到Student表中的元组处理完为止。

9.1 关系数据库系统的查询处理

（4）Hash Join方法

- 把连接属性作为hash码，用同一个hash函数把R和S中的元组散列到同一个hash文件中
- **步骤：**
  - 划分阶段：
    - ✓对包含较少元组的表(比如R)进行一遍处理
    - ✓把它的元组按hash函数分散到hash表的桶中
  - 试探阶段：也称为连接阶段
    - ✓对另一个表(S)进行一遍处理
    - ✓把S的元组散列到适当的hash桶中
    - ✓把元组与桶中所有来自R并与之相匹配的元组连接起来
- 算法前提：假设两个表中较小的表在第一阶段后可以完全放入内存的hash桶中。
- 以上的算法思想可以推广到更加一般的多个表的连接算法上。

第9章 关系查询处理与查询优化

9.2 关系数据库系统的查询优化

- ✓ 查询优化在关系数据库系统中有着非常重要的地位
- ✓ 查询优化是影响RDBMS性能的关键因素
- ✓ 由于关系表达式的语义级别很高，使关系系统可以从关系表达式中分析查询语义，提供了执行查询优化的可能性

9.2 关系数据库系统的查询优化

9.2.1 查询优化的必要性

[例] 求选修了2号课程的学生姓名。用SQL表达：

```
SELECT Student.Sname FROM Student，SC
    WHERE Student.Sno=SC.Sno AND C.Cno='2'；
```

- 假定学生-课程数据库中有1000个学生记录，10000个选课记录
- 其中选修2号课程的选课记录为50个
- 有多种等价的关系代数表达式来完成这一查询：

$Q_1 = \pi_{Sname}(\sigma_{Student.Sno=SC.Sno \wedge Sc.Cno='2'}(Student \times SC))$

$Q_2 = \pi_{Sname}(\sigma_{Sc.Cno='2'}(Student \bowtie SC))$

$Q_3 = \pi_{Sname}(Student \bowtie \sigma_{Sc.Cno='2'}(SC))$

9.2 关系数据库系统的查询优化

- **设一个块能装10个Student元组或100个SC元组，在内存中存放5块Student元组和1块SC元组，每秒读写20块，内存处理时间忽略。**

❖ **第一种情况** $Q_1=\pi_{Sname}(\sigma_{Student.Sno=SC.Sno \wedge Sc.Cno='2'} Student \times SC))$

1. 计算广义笛卡尔积

✓ 读取总块数为：$\frac{1000}{10}+\frac{1000}{10\times5}\times\frac{10000}{100}$ =100+20×100=2100块 => 105s

✓ 连接后的元组数为$10^3\times10^4=10^7$。设每块能装10个元组，则写出这些块要用$10^6/20=5\times10^4$s

2. 选择操作

✓ 读取中间文件花费的时间(同写中间文件一样)需5×10^4s

✓ 满足条件的元组假设仅50个，均可放在内存

本策略总耗时（投影处理时间忽略）：$105+2\times5\times10^4\approx10^5$s （27.8小时）

9.2 关系数据库系统的查询优化

❖ **第二种情况** $Q_2=\pi_{Sname}(\sigma_{Sc.Cno='2'}(Student \bowtie SC))$

1. 计算自然连接

✓ 执行自然连接，读取Student和SC表的策略不变，总的读取块数仍为2100块花费105 s

✓ 自然连接的结果比第一种情况大大减少，为10^4个

✓ 写出这些元组时间为$10^4/10/20=50$s，为第一种情况的千分之一

2. 读取中间文件块，执行选择运算，花费时间也为50s。

3. 把第2步结果投影输出。

本策略总的执行时间 ≈ 105+50+50≈205s

9.2 关系数据库系统的查询优化

❖ **第三种情况**

$$Q_3=\pi_{Sname}(\text{Student} \bowtie \sigma_{Sc.Cno='2'}(SC))$$

1. 先对SC表作选择运算，只需读一遍SC表，存取100块花费时间为5s，因为满足条件的元组仅50个，不必使用中间文件。
2. 读取Student表，把读入的Student元组和内存中的SC元组作连接。也只需读一遍Student表共100块，花费时间为5s。
3. 把连接结果投影输出

本策略总的执行时间 ≈ 5+5≈10s

9.2 关系数据库系统的查询优化

9.2.2 代数优化

- 代数优化策略：通过关系代数表达式的等价变换来提高查询效率
- 关系代数表达式的等价：指用相同的关系代替两个表达式中相应的关系所得到的结果是相同的
- 两个关系表达式E_1和E_2是等价的，可记为E_1≡E_2

9.2 关系数据库系统的查询优化

■ **关系代数表达式的等价变换规则**

1. 连接、笛卡尔积交换律

设E_1和E_2是关系代数表达式，F是连接运算的条件，则有

$$E_1 \times E_2 \equiv E_2 \times E_1$$
$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$$
$$E_1 \underset{F}{\bowtie} E_2 \equiv E_2 \underset{F}{\bowtie} E_1$$

2. 连接、笛卡尔积的结合律

设E_1，E_2，E_3是关系代数表达式，F_1和F_2是连接运算的条件，则有

$$(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$$
$$(E_1 \bowtie E_2) \bowtie E_3 \equiv E_1 \bowtie (E_2 \bowtie E_3)$$
$$(E_1 \underset{F_1}{\bowtie} E_2) \underset{F_2}{\bowtie} E_3 \equiv E_1 \underset{F_1}{\bowtie} (E_2 \underset{F_2}{\bowtie} E_3)$$

9.2 关系数据库系统的查询优化

3. 投影的串接定律

$$\Pi_{A_1,A_2,\cdots,A_n}(\Pi_{B_1,B_2,\cdots,B_m}(E)) \equiv \Pi_{A_1,A_2,\cdots,A_n}(E)$$

这里，E是关系代数表达式，A_i(i=1，2，…，n)，B_j(j=1，2，…，m)是属性名，且{A_1，A_2，…，A_n}构成{B_1，B_2，…，B_m}的子集。

4. 选择的串接定律

$$\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_1 \wedge F_2}(E)$$

这里，E是关系代数表达式，F_1、F_2是选择条件。

选择的串接律说明选择条件可以合并。这样一次就可检查全部条件。

9.2 关系数据库系统的查询优化

5. 选择与投影操作的交换律

$$\sigma_F(\Pi_{A_1,A_2,\cdots,A_n}(E)) \equiv \Pi_{A_1,A_2,\cdots,A_n}(\sigma_F(E))$$

这里，选择条件 F 只涉及属性 A_1，…，A_n。

若 F 中有不属于 A_1，…，A_n 的属性 B_1，…，B_m，则有更一般的规则：

$$\Pi_{A_1,A_2,\cdots,A_n}(\sigma_F(E)) \equiv \Pi_{A_1,A_2,\cdots,A_n}(\sigma_F(\Pi_{A_1,A_2,\cdots,A_n,B_1,B_2,\cdots,B_m}(E)))$$

6. 选择与笛卡尔积的交换律

如果F中涉及的属性都是E_1中的属性，则

$$\sigma_F(E_1 \times E_2) \equiv \sigma_F(E_1) \times E_2$$

如果$F = F_1 \wedge F_2$，并且 F_1 只涉及 E_1 中的属性，F_2 只涉及 E_2 中的属性，则由上面的等价变换规则1，4，6可推出：

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2)$$

若F_1只涉及E_1中的属性，F_2涉及E_1和E_2两者的属性，则仍有

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2)$$

它使部分选择在笛卡尔积前先做。

9.2 关系数据库系统的查询优化

7. 选择与并的分配律

设$E = E_1 \cup E_2$，E_1，E_2有相同的属性名，则：

$$\sigma_F(E_1 \cup E_2) \equiv \sigma_F(E_1) \cup \sigma_F(E_2)$$

8. 选择与差运算的分配律

若E_1与E_2有相同的属性名，则：

$$\sigma_F(E_1 - E_2) \equiv \sigma_F(E_1) - \sigma_F(E_2)$$

9. 选择对自然连接的分配律

$$\sigma_F(E_1 \bowtie E_2) \equiv \sigma_F(E_1) \bowtie \sigma_F(E_2)$$

F只涉及E_1与E_2的公共属性

9.2 关系数据库系统的查询优化

10. 投影与笛卡尔积的分配律

设E_1和E_2是两个关系表达式，A_1，…，A_n是E_1的属性，B_1，…，B_m是E_2的属性，则：

$$\pi_{A_1,A_2,\cdots,A_n,B_1,B_2,\cdots,B_m}(E_1 \times E_2) \equiv \pi_{A_1,A_2,\cdots,A_n}(E_1) \times \pi_{B_1,B_2,\cdots,B_m}(E_2)$$

11. 投影与并的分配律

设E_1和E_2有相同的属性名，则：

$$\pi_{A_1,A_2,\cdots,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,A_2,\cdots,A_n}(E_1) \cup \pi_{A_1,A_2,\cdots,A_n}(E_2)$$

9.2 关系数据库系统的查询优化

■ **查询树的启发式优化**

典型的启发式规则：

1. 选择运算应尽可能先做。在优化策略中这是最重要、最基本的一条；
2. 把投影运算和选择运算同时进行；

   如有若干投影和选择运算，并且它们都对**同一个关系**操作，则可以在扫描此关系的同时完成所有的这些运算以避免重复扫描关系
3. 把投影同其前或其后的双目运算结合起来；
4. 把某些选择同在它前面要执行的笛卡尔积结合起来成为一个连接运算；
5. 找出公共子表达式。

   如果这种重复出现的子表达式的结果不是很大的关系，并且从外存中读入这个关系比计算该子表达式的时间少得多，则可以先计算一次公共子表达式并把结果写入中间文件。当查询的是视图时，定义视图的表达式就是公共子表达式的情况。

9.2 关系数据库系统的查询优化

9.2.3 物理优化

- ✓ 代数优化改变查询语句中操作的次序和组合，不涉及底层的存取路径；
- ✓ 对于一个查询语句有许多存取方案，它们的执行效率不同，仅仅进行代数优化是不够的；
- ✓ 物理优化就是要选择高效合理的操作算法或存取路径，求得优化的查询计划。

- □ 基于规则的启发式优化
- □ 基于代价估算的优化
- □ 两者结合的优化方法

9.2 关系数据库系统的查询优化

■ **基于启发式规则的存取路径选择优化**

■ **选择操作的启发式规则**

1. 小关系，使用全表顺序扫描，即使选择列上有索引。
2. 大关系，启发式规则有：
   1）选择条件是主码＝值的查询，可以选择主码索引。
   2）选择条件是非主属性＝值的查询，且选择列上有索引：
      - 估算查询结果的元组数目，若比例较小(<10%)可以使用索引，否则全表扫描

   3）选择条件是非等值查询或者范围查询，且选择列上有索引：
      - 方法同上！

9.2 关系数据库系统的查询优化

4）对于使用 AND 的组合条件

- **如果有涉及这些属性的组合索引，优先采用组合索引扫描方法**
- **如果某些属性上有一般的索引，则可以采用索引扫描方法，否则使用全表顺序扫描。**

5）对于使用 OR 的组合条件，一般使用全表顺序扫描。

9.2 关系数据库系统的查询优化

■ **连接操作的启发式规则：**

1. 如果 2 个表都已经按照连接属性排序： 选用**排序-合并**方法
2. 如果一个表在连接属性上有索引： 选用**索引连接**方法
3. 如果上面2个规则都不适用，其中一个表较小： 选用Hash join方法
4. 选用**嵌套循环**方法，并选择其中较小的表，确切地讲是占用的块数(b)较少的表，作为外表(外循环的表)。

- **本章作业**
  - 习题第3题
  - 阐述DBMS查询处理的步骤及内容。